Cours d'Informatique 2

1ère année Sciences de la matière

2024/2025, Semestre 2

SOUIOU .W

Département de Sciences de la matière,

Université Badji Mokhtar ANNABA

souiou@yahoo.fr

Objectif et plan du cours

- **Objectif:**
  - Apprendre les concepts de base de la programmation
  - Etre capable de mettre en œuvre ces concepts pour analyser des problèmes simples et écrire les programmes correspondants
- **Plan:**
  - Généralités (matériel d'un ordinateur, systèmes d'exploitation, langages de programmation, …)
  - Fortran (un langage de programmation)

Informatique?

- Techniques du traitement **automatique** de l'**information** au moyen des ordinateurs

- Eléments d'un système informatique

Matériel: Principaux éléments d'un PC

- Unité centrale (le boîtier)
  - Processeur ou CPU (*Central Processing Unit*)
  - Mémoire centrale
  - Disque dur, lecteur disquettes, lecteur CD-ROM
  - Cartes spécialisées (cartes vidéo, réseau, ...)
  - Interfaces d'entrée-sortie (Ports série/parallèle, …)
- Périphériques
  - Moniteur (l'écran), clavier, souris
  - Modem, imprimante, scanner, …

Qu’est ce qu’un système d’exploitation?

- Ensemble de programmes qui gèrent le matériel et contrôlent les applications
  - Gestion des périphériques (affichage à l'écran, lecture du clavier, pilotage d’une imprimante, …)
  - Gestion des utilisateurs et de leurs données (comptes, partage des ressources, gestion des fichiers et répertoires, …)
  - Interface avec l’utilisateur (textuelle ou graphique): Interprétation des commandes
  - Contrôle des programmes (découpage en taches, partage du temps processeur, …)

Langages informatiques

- Un langage informatique est un outil permettant de donner des ordres (**instructions**) à la machine
  - A chaque instruction correspond une action du processeur
- Intérêt : écrire des **programmes** (suite consécutive d'instructions) déstinés à effectuer une tache donnée
  - Exemple: un programme de gestion de comptes bancaires
- Contrainte: être compréhensible par la machine

Langage machine

- Langage **binaire**: l'information est exprimée et manipulée sous forme d'une suite de bits
- Un **bit** (*binary digit*) = 0 ou 1 (2 états électriques)
- Une combinaison de 8 bits= 1 **Octet** ➔ $2^8 = 256$ possibilités qui permettent de coder tous les caractères alphabétiques, numériques, et symboles tels que ?,*,&, …
  - Le code **ASCII** (*American Standard Code for Information Interchange*) donne les correspondances entre les caractères alphanumériques et leurs représentation binaire, Ex. A= 01000001, ?=00111111
- Les opérations logiques et arithmétiques de base (addition, multiplication, …) sont effectuées en binaire

L'assembleur

- Problème: le langage machine est difficile à comprendre par l'humain

- Idée: trouver un langage compréhensible par l'homme qui sera ensuite converti en langage machine
  - **Assembleur** (1er langage): exprimer les instructions élémentaires de façon symbolique

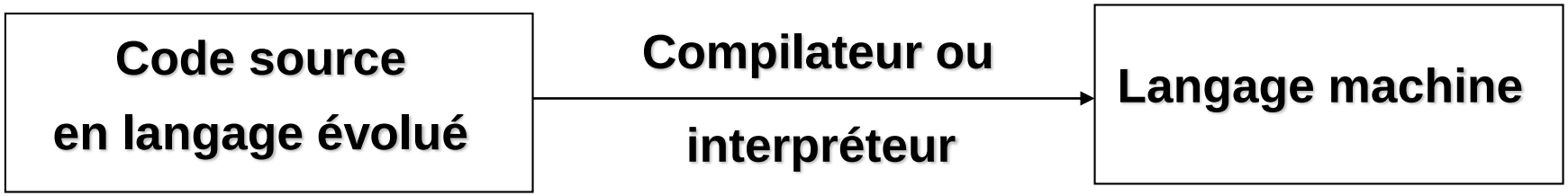
ADD A, 4
LOAD B → traducteur → langage machine
MOV A, OUT
...

  - +: déjà plus accessible que le langage machine
  - -: dépend du type de la machine (n'est pas ***portable***)
  - -: pas assez efficace pour développer des applications complexes

⇒ **Apparition des langages évolués**

Langages haut niveau

- Intérêts multiples pour le haut niveau:
  - proche du langage humain «anglais» (compréhensible)
  - permet une plus grande portabilité (indépendant du matériel)
  - Manipulation de données et d'expressions complexes (réels, objets, a*b/c, …)
- Nécessité d'un traducteur (compilateur/interpréteur), exécution plus ou moins lente selon le traducteur



Compilateur/interpréteur

- Compilateur: traduire le programme entier une fois pour toutes

  - + plus rapide à l'exécution
  - + sécurité du code source
  - - il faut recompiler à chaque modification

- Interpréteur: traduire au fur et à mesure les instructions du programme à chaque exécution

  - + exécution instantanée appréciable pour les débutants
  - - exécution lente par rapport à la compilation

Langages de programmation:

- Deux types de langages:
  - Langages procéduraux
  - Langages orientés objets
- Exemples de langages:
  - **Fortran, Cobol, Pascal, C, …**
  - **C++, Java, …**
- Choix d'un langage?

Etapes de réalisation d'un programme

Enoncé du problème

↓ *Spécification*

Cahier des charges

↓ *Analyse*

Algorithme

↓ *Traduction en langage*

Programme source

↓ *Compilation*

Programme exécutable

↓ *Tests et modifications*

Version finale et résultats

La réalisation de programmes passe par l'écriture d'algorithmes

$\Rightarrow$ D'où l'intérêt de l'**Algorithmique**

PROGRAMME FORTRAN

- FORTRAN est un langage compilé, un code exécutable peut être composé de un ou plusieurs fichiers ASCII.

❖ **Etapes de création d'un programme fortran:**

- Edition ⟶ Fichier source
- Compilation ⟶ Fichier objet
- Edition de lien ⟶ Fichier exécutable
- Exécution

❖ **Constitution d'un programme Fortran:**

1. L'alphabet
2. La grammaire:

   Mots clés, Opérandes, Séparateurs

3. **Les opérateurs** :

- **L’affectation**
- **Les opérateurs arithmétique :**
  - Exponentiation
  - Multiplication
  - Division
  - Addition
  - Soustraction
- **Les Opérateurs relationnels:**
  - .LT. Strictement plus petit que
  - .GT. Strictement plus grand que

- .LE. Plus petit ou égal à
- .GE. Plus grand ou égal à
- .EQ. Egal à
- .NE. Non égal à

➢ **Les opérateurs logiques:**

.NOT. Négation logique

.AND. ET logique

.OR. OU logique

DONNEES

- TYPE DE DONNEES:

Entier ⟶ INTEGER

Réel ⟶ REAL

Complexe ⟶ COMPLEX

Chaine de caractère ⟶ CHARACTER*len

Logique ⟶ LOGICAL

DECLARATION DE DONNEES:

déclarations explicites:

Exemple: INTEGER nbr, cpt

REAL x,y

DONNEES

- **Déclaration implicite:**

Une des particularités de fortran est qu'il existe des déclarations de données implicites suivant le nom des variables. Ainsi, si aucune spécification contraire n'est rencontrée, les déclarations implicites sont:

- Toute variable commençant par I,J,K,L,M,N est de type INTEGER.
- Toute variable commençant par une autre lettre est de type REAL.

Cependant, il est possible de modifier cette règle; pour ce faire, on utilise l'instruction **IMPLICITE NONE** placée obligatoirement en tète de programme ou sous-programme.

ENTRÉES/SORTIES

1- unité logique:

En fortran, tout périphérique est repéré par un entier, on parle alors d'unité logique.

- L' unité logique représentant le clavier est: 5
- L' unité logique représentant le clavier est: 6

Mais , si l'on ne veut pas retenir ces numéros, on peut placer le caractère (*) .

2- la syntaxe générale des instructions d'entrée /sorties:

Read		
Ou	**(lu,etq)**	**listv**
Write		

etq Format (lists)

ENTRÉES/SORTIES

- **Instruction Read:**

Exp: READ (5,100) A,B

100 FORMAT (….,….)

ou bien

READ *, A,B

READ (*,*) A,B

- **Instruction Write:**

EXP: WRITE (6,50) tab

50 format (….,….)

ou bien

PRINT *, tab

WRITE (*,*)tab

STRUCTURES DE CONTROLE

1- CONTROLES CONDITIONNELS:

- **INSTRUCTION IF:**

- IF (CONDITION) THEN
  INSTRUCTIONS
  ENDIF
- IF (CONDITION) THEN
  INSTRUCTION 1
  ELSE
  INSTRUCTION 2
  ENDIF
- IF (CONDITION 0) THEN
  INSTRUCTION 0
  ELSEIF (CONDITION 1) THEN
  INSTRUCTION 1
  ELSE
  INSTRUCTION N
  ENDIF

STRUCTURES DE CONTROLE

2- BOUCLES DE CONTRÔLE:

- **INSTRUCTION DO:**

```
DO  icpt= idebut, ifin, [ipas]
   instructions
ENDDO
```

- **INSTRUCTION DOWHILE:**

```
DOWHILE (condition)
  instructions
ENDDO
```

TABLEAUX

- **Instruction Dimension:**

L'Instruction Dimension permet de définir la taille d'un tableau dont le type a été déclaré précédemment de façon explicite ou implicite.

Exemple:

```
Real        X
Complexe    Y
Integer     Tab(100)
Dimension   X( 10) ,Y  (100)
```